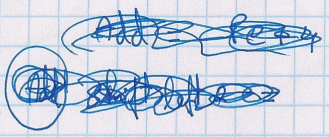
4.7. (5) Para a ALU ou para a unidade de Add quais os seus valores de Input?



ALU = Sign extend & - 3

ADD Branch = PC + 4
shift left 2

ADD PC = PC
4

4.7.6

Políticas de Registo

RR1 - (3)

RR2 - (2)

Write register - entra o valor zero logo é o (2)

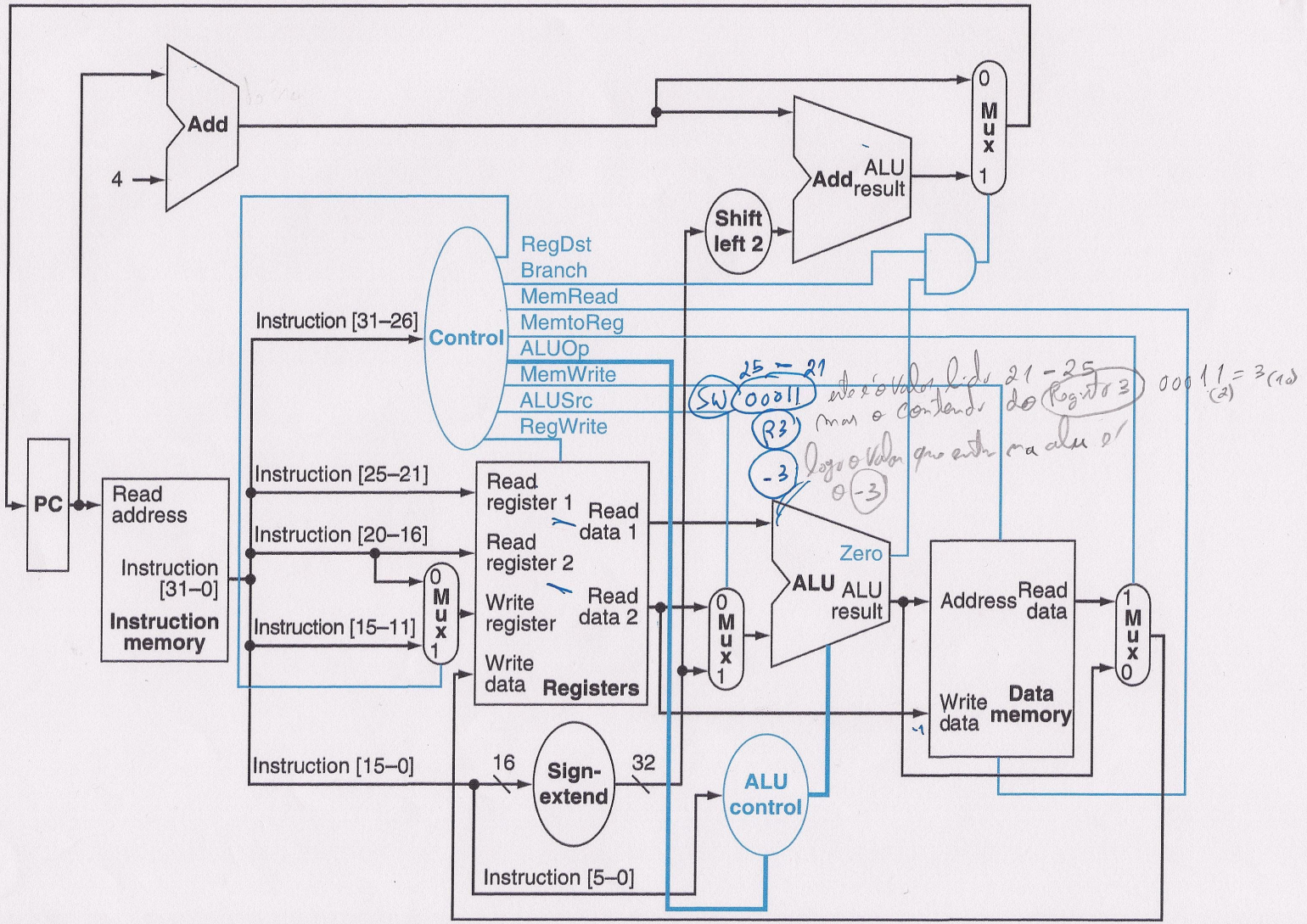
RegWrite bit de controlo dos registos é (0)

Write Data:

Pode ser uma coisa ou outra pois o valor que vem do multiplexer é Don't Care na instrução SW

4.7.5

4.7.6


Add
4
0
M
u
x
1
Add
ALU result
Shift left 2
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
PC
Read address
Instruction [31–0]
Instruction memory
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
ALU control
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
25 – 21
SW
00011
R3
-3
Logo o valor que entra na alu é o -3
00011 (2) = 3 (10)

Exercícios 4ª aula

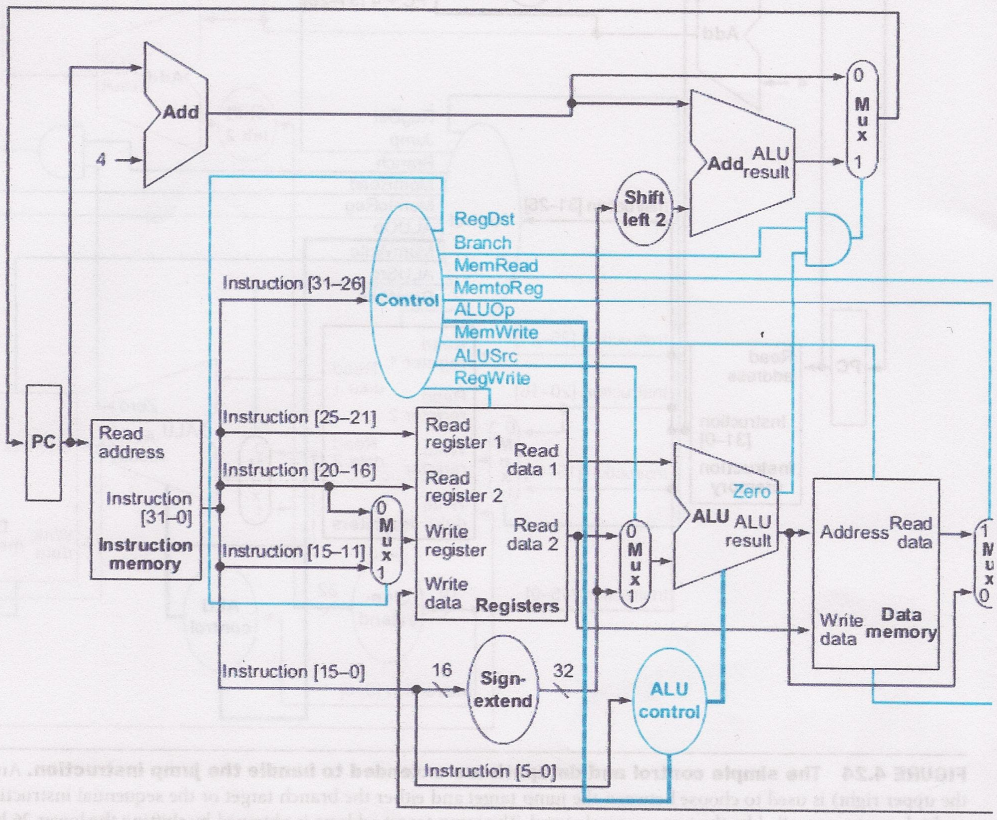


FIGURE 4.17 The simple datapath with the control unit. The input to the control unit is the 6-bit opcode field from t The outputs of the control unit consist of three 1-bit signals that are used to control multiplexors (RegDst, ALUSrc, and Mei signals for controlling reads and writes in the register file and data memory (RegWrite, MemRead, and MemWrite), a 1-bit determining whether to possibly branch (Branch), and a 2-bit control signal for the ALU (ALUOp). An AND gate is used t branch control signal and the Zero output from the ALU; the AND gate output controls the selection of the next PC. Notice tha a derived signal, rather than one coming directly from the control unit. Thus, we drop the signal name in subsequent figures.

4.3 When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are starting with a datapath from Figure 4.2, where I-Mem, Add, Mux, ALU, Regs, D-Mem, and Control blocks have latencies of 400 ps, 100 ps, 30 ps, 120 ps, 200 ps, 350 ps, and 100 ps, respectively, and costs of 1000, 30, 10, 100, 200, 2000, and 500, respectively.

Consider the addition of a multiplier to the ALU. This addition will add 300 ps to the latency of the ALU and will add a cost of 600 to the ALU. The result will be 5% fewer instructions executed since we will no longer need to emulate the MUL instruction.

4.3.1 [10] <§4.1> What is the clock cycle time with and without this improvement?

4.3.2 [10] <§4.1> What is the speedup achieved by adding this improvement?

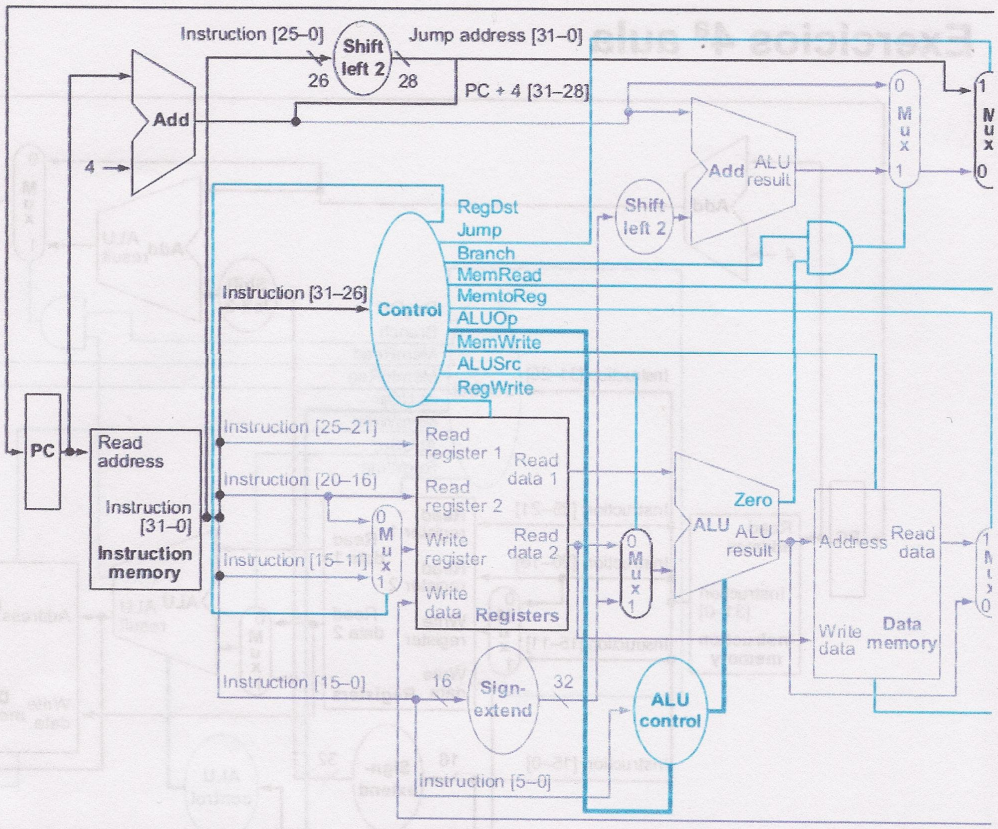


FIGURE 4.24 The simple control and datapath are extended to handle the jump instruction. An additional
the upper right) is used to choose between the jump target and either the branch target or the sequential instruction followin
multiplexor is controlled by the jump control signal. The jump target address is obtained by shifting the lower 26 bits of the ju
left 2 bits, effectively adding 00 as the low-order bits, and then concatenating the upper 4 bits of PC + 4 as the high-order bits,
32-bit address.

4.7 In this exercise we examine in detail how an instruction is executed in a single-cycle datapath. Problems in this exercise refer to a clock cycle in which the processor fetches the following instruction word:

`10101100011000100000000000010100`.

Assume that data memory is all zeros and that the processor's registers have the following values at the beginning of the cycle in which the above instruction word is fetched:

r0	r1	r2	r3	r4	r5	r6	r8	r12	r31
0	-1	2	-3	-4	10	6	8	2	-16

4.7.1 [5] <§4.4> What are the outputs of the sign-extend and the jump "Shift left 2" unit (near the top of Figure 4.24) for this instruction word?

4.7.2 [10] <§4.4> What are the values of the ALU control unit's inputs for this instruction?

4.7.3 [10] <§4.4> What is the new PC address after this instruction is executed? Highlight the path through which this value is determined.

4.7.4 [10] <§4.4> For each Mux, show the values of its data output during the execution of this instruction and these register values.

4.7.5 [10] <§4.4> For the ALU and the two add units, what are their data input values?

4.7.6 [10] <§4.4> What are the values of all inputs for the "Registers" unit?

4.X. Considerando a figura 4.17, assuma que vai implementar duas novas instruçõe

- `jal` (jump and link)
- `seq` (set if equal)

Para cada uma, diga:

a) quais a unidades funcionais existentes usadas pela instrução

b) que unidades funcionais adicionais são necessárias

c) que novos sinais de controlo são necessários

d) quais os valores de todos os sinais de controlo (para a saída do ALU control, bas a operação efectuada pela ALU)

Última alteração: Quinta, 9 Outubro 2014, 14:52

Exercícios 5ª Aula

4.8 In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
250ps	350ps	150ps	300ps	200ps

Also, assume that instructions executed by the processor are broken down as follows:

alu	beq	lw	sw
45%	20%	20%	15%

4.8.1 [5] <§4.5> What is the clock cycle time in a pipelined and non-pipelined processor?

4.8.2 [10] <§4.5> What is the total latency of an LW instruction in a pipelined and non-pipelined processor?

4.8.3 [10] <§4.5> If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

4.8.4 [10] <§4.5> Assuming there are no stalls or hazards, what is the utilization of the data memory?

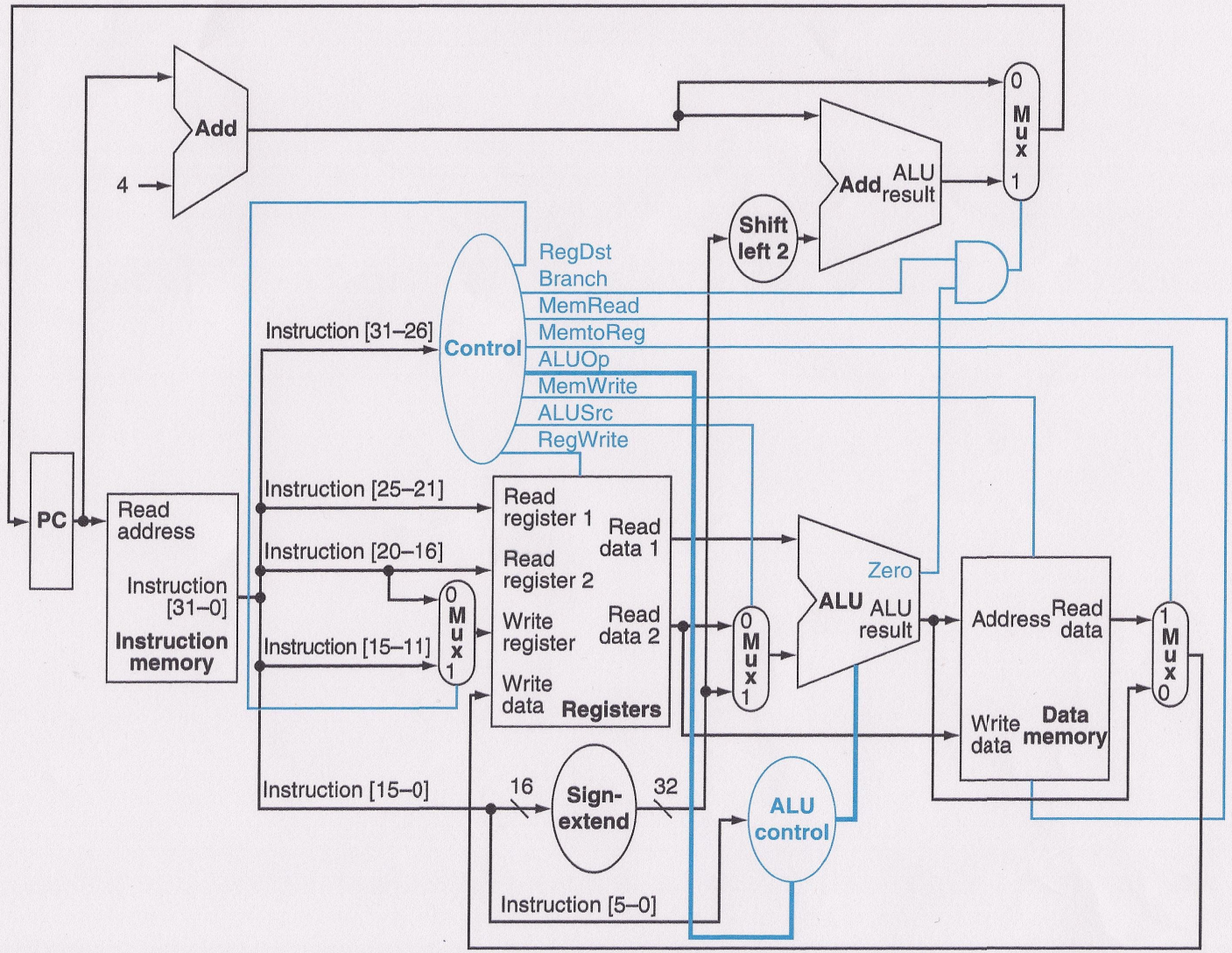
4.8.5 [10] <§4.5> Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?

Calcular o CPI da implementação pipelined (5 andares) para
- 1 instrução
- 10^6 instruções (sem atrasos)
- 10^6 instruções, com uma em cada 5 instruções a ser atrasada um ciclo.

Para o código abaixo:
+Identifique todas as dependências (de dados) existentes.
(Cuidado com os saltos.)
+Identifique os conflitos de dados.
+ Introduza os *nop* necessários para eliminar os conflitos num *pipeline* sem *forwarding*.
+ Introduza os *nop* necessários para eliminar conflitos no *pipeline* com *forwarding*.

```
1:          or $3, $0, $0
2:          or $8, $5, $0
3:  início: beq $8, $0, fim
4:          addiu $4, $4, -4
5:          lw $9, 0($4)
6:          add $3, $3, $9
7:          beq $0, $0, início
8:          addi $8, $8, -1
9:      fim: sub $3, $7, $3
```

Última alteração: Segunda, 12 Outubro 2015, 20:13


Add
4
0
M
u
x
1
Add
ALU result
Shift left 2
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
ALU control
Zero
ALU
ALU result
Address
Read data
Write data
Data memory

4.8

IF	ID	Ex	MEM	WB
250 Ps	350 Ps	150 Ps	300 Ps	200 Ps

IF	ID	Ex	MEM	WB	
	IF	ID	Ex	MEM	WB

→ Pipelined

IF	ID	Ex	MEM	WB	
	IF	ID	Ex	MEM	WB

→ non Pipelined

4.8.1) non pipelined : 250 + 350 + 150 + 300 + 200 = 1250 Ps

pipelined : 350

4.8.2) non pipelined : 1250 Ps

Pipelined : 5 x 350 = 1750

4.8.3)

IF	ID1	ID2	Ex	MEM	WB
250	175	175	150	300	200

↓ cada com 6 estágios do pipelined, o novo clock cycle começa logo aos 300 Ps

4.8.4)

ALU	beq	lw	sw
45%	20%	20%	15%

LW + SW = 35 %

quais as que utilizam a memória? R: Não o LW e SW

4.8.5) quais as instruções que escrevem nos registos?

R: ALU e LW = 65 %

Exercício que não está no livro:

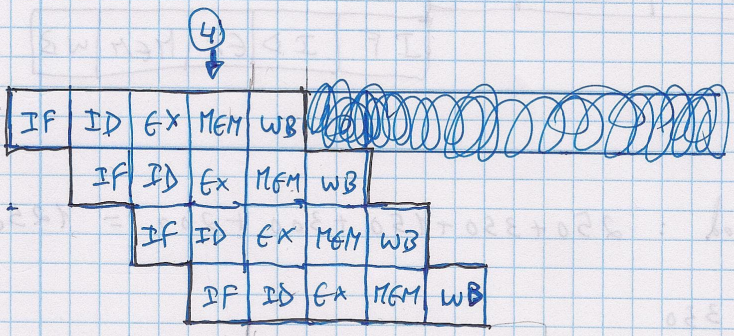
Calcular o CPI da implementação pipelined (5 andares) para:

$$CPI = \frac{nº\ ciclos}{nº\ instruções}$$

(A) - 1 instrução

(B) - 10^6 instruções (sem atrasos)

(C) - 10^6 instruções, com uma em cada 5 instruções a ser atrasada um ciclo



(A) = $CPI = \frac{5}{1} = 5$ // necessitamos de 5 ciclos para executar 1 instrução

(B) $CPI = \frac{10^6 + 4}{10^6} \simeq 1$ //

O 4 significa que é o limite do pipelined até encher e iniciarem todas as instruções

(C) $CPI = \frac{10^6 + 4 + \left(\frac{10^6}{5}\right)}{10^6} \simeq 1,2$ //

Para o código seguinte Identifique todas as dependências de dados existentes. (Cuidado com os saltos)

nº instruções	Código
1:	or $3, $0, $0
2:	or $8, $5, $0
3:	Inicio: beq $8, $0, Fim
4:	addi $4, $4, -4
5:	lw $9, 0($4)
6:	add $3, $3, $9
7:	beq $0, $0, Inicio
8:	addi $8, $8, -1
9:	Fim: sub $3, $7, $3

a) 3 → 2 : $8

dependência de registo $8 na 2ª instrução

3 → 2 : $8

5 → 4 : $4

6 → 5 : $9

8 → 2 : $8

3 → 8 : $8

6 → 1 : $3

Durante a repetição do ciclo existe uma dependência dentro de si própria 6 → 6 : $3

8 → 8 : $8

4 → 4 : $4

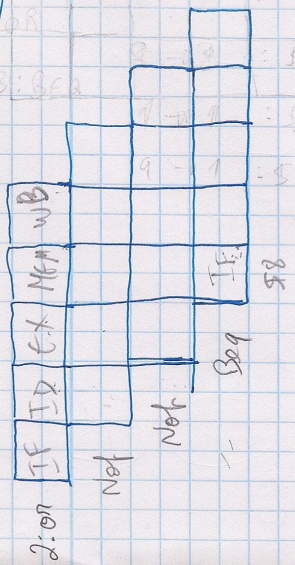
9 → 1 : $3

9 → 6 : $3

4 → 4 : $4

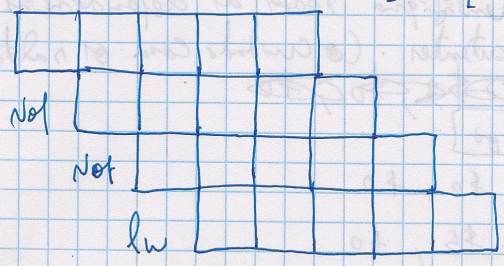
b) Identifique os Conflitos de dados

3 → 2 2 ciclos 2 Nop



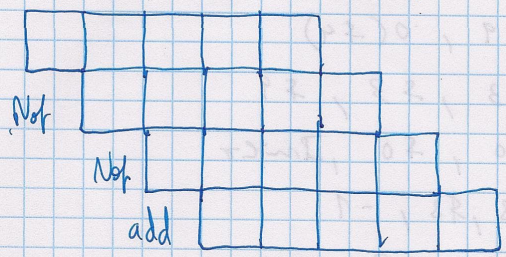
5 → 4 necessitamos 2 NoP

addiu



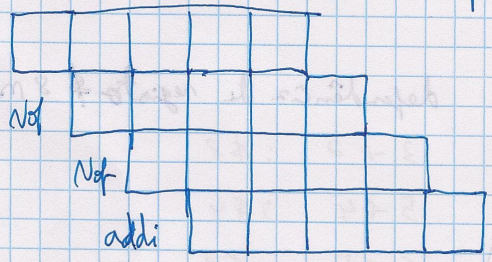
6 → 5 2 Nop

lw



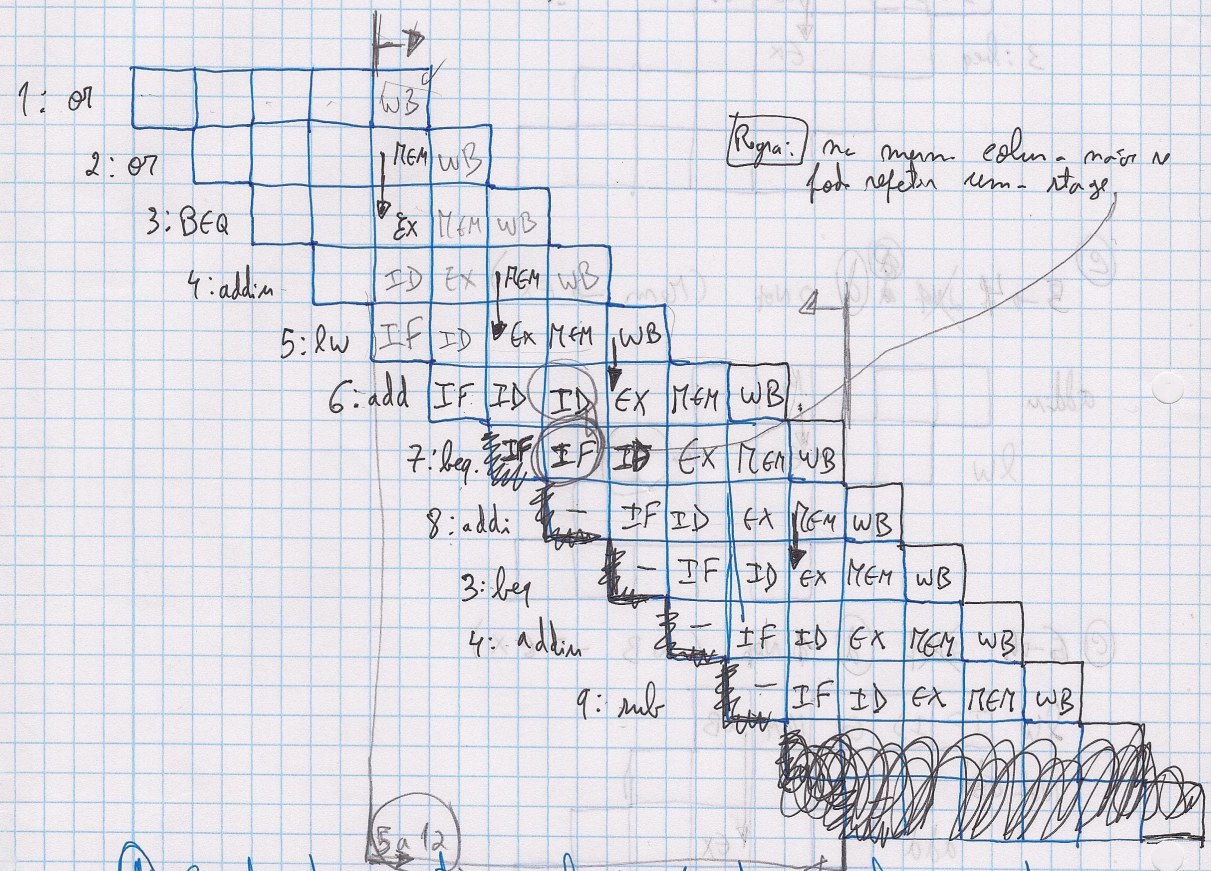
3 → 8 2 Nop

beq



(c) Introduza os nop necessários para eliminar os conflitos num ficheiro sem forwarding

② Apresente a evolução do estado do pipeline com forwarding durante a execução do código, sendo a instrução 3 executada 2 vezes. Indique todos os atrasos introduzidos e todos os pontos onde foi necessário fazer forwarding de algum valor.



③ Considerando somente os ciclos 5 a 12 da simulação feita na alínea anterior, calcule a percentagem dos ciclos de relógio em que todos os andares do pipeline são efectivamente usados por alguma instrução.

R: a percentagem de ciclos que usam o pipeline na totalidade é zero %

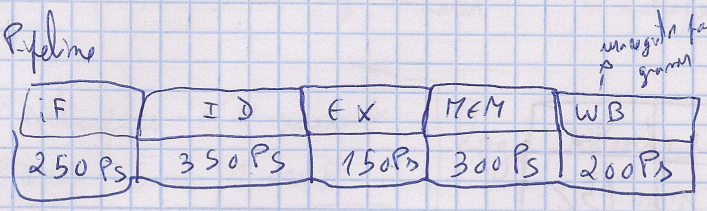
(g) Modifique o código de modo a que a sua execução não possa dar origem a interbloqueios.

(Resposta) Trocando a linha 6 pela linha 8

4.8

Latência - Tempo de resposta de cada fase do Pipeline

Pipeline



- No Non Tempo é 350 PS logo este é o período de relógio mais, ou seja a duração do C.do de relógio para uma implementação Pipelined

- O para implementação não pipelined seria a soma 1250 PS este seria o c.do de relógio

lw / não Pipelined } =
e Pipelined } uma instrução lw eleva 5x350 PS a ser executada

Para melhorar este Pipelined teria de se dividir o ID em 2 e o mais tempo a de pipelined faria a ser MEM 300PS melhorando a performance

esta alteração de ID teria de ser feita com hardware

$$Speedup = \frac{\text{Desempenho do } CPU_A}{\text{Desempenho do } CPU_B}$$

$$T = \frac{n^\circ \text{ ciclos}}{f} = \frac{n^\circ \text{ instruções} \times CPI}{f} = n^\circ \text{ instruções} \times CPI \times t$$

alu	beq	lw	sw
45%	20%	20%	15%

CPI - 1 instrução = 5 ciclos

CPI - 10^6 + 4 ciclos = CPI = 1

CPI = 10^6 x 1 em cada 5 instruções é atribuído 1 ciclo =

$10^6 \times 1.2 + 4 =$

= CPI = 1,2

1

1 in Instrução	ciclos
1	5 ciclos
2	5 + 1 ciclo
3	6 + 1 = 5 + 2
4	7 + 1 = 5 + 3
5	8 + 1 = 5 + 4
6	9 + 1 = 6 + 4

Alínea 2 — full-forwarding — uso de instruções já existentes em (A)

	1	2	3	4	5	6	7	8	9	10
(A)	IF	ID	EX	MEM ($1)	WB ($1)					
(B)		IF	ID	EX	MEM ($2)	WB				
(C)			IF	ID	EX	MEM	WB			

Alínea 3 — Só Mem-Ex (ou Alu-Alu)
Só forwarding no Mem & Ex

	1	2	3	4	5	6	7	8
(A)	IF	ID	EX	MEM	WB			
(B)		IF	ID	EX				
(C)			IF					

Nota: forwarding mais em conflito de dados

Exercícios 6ª Aula

(Continua da aula anterior)

Na resolução deste exercício, assuma que a decisão sobre se um salto condicional é ou não efectuado é tomada no andar EX e que é usado um delay slot. Para o código abaixo:

1. or $3, $0, $0
2. or $8, $5, $0
3. início: beq $8, $0, fim
4. addiu $4, $4, -4
5. lw $9, 0($4)
6. add $3, $3, $9
7. beq $0, $0, início
8. addi $8, $8, -1
9. fim: sub $3, $7, $3

a) Identifique todas as dependências (de dados) existentes. (Cuidado com os saltos.)
b) Identifique os conflitos de dados.
c) Introduza os nop necessários para eliminar os conflitos num pipeline sem forwarding.
d) Introduza os nop necessários para eliminar conflitos no pipeline com forwarding.
e) Apresente a evolução do estado do pipeline comforwarding durante a execução do código, sendo a instrução 3 executada 2 vezes. Indique todos os atrasos introduzidos e todos os pontos onde foi necessário fazer forwarding de algum valor.
f) Considerando somente os ciclos 5 a 12 da simulação feita na alínea anterior, calcule a percentagem dos ciclos de relógio em que todos os andares do pipeline são efectivamente usados por alguma instrução.
g) Para o ciclo da execução em que o conteúdo dopipeline é o indicado abaixo, qual é o valor dos sinais de controlo
usados em cada um dos andares? (ver figura-04-51 dos recursos)

Andar	Instrução
IF	add $3, $3, $9
ID	lw $9, 0($4)
EX	addiu $4, $4, -4
MEM	beq $8, $0, fim
WB	or $8, $5, $0

h) Modifique o código de modo a que a sua execução se possa dar sem a introdução de qualquer atraso.

Última alteração: Terça, 20 Outubro 2015, 19:30

Exercicios 7a Aula

4.10 In this exercise, we examine how resource hazards, control hazards, and Instruction Set Architecture (ISA) design can affect pipelined execution. Problems in this exercise refer to the following fragment of MIPS code:

```
sw  r16,12(r6)
lw  r16,8(r6)
beq r5,r4,Label # Assume r5!=r4
add r5,r1,r4
slt r5,r15,r4
```

Assume that individual pipeline stages have the following latencies:

IF	ID	EX	MEM	WB
200ps	120ps	150ps	190ps	100ps

4.10.1 [10] <§4.5> For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we only have one memory (for both instructions and data), there is a structural hazard every time we need to fetch an instruction in the same cycle in which another instruction accesses data. To guarantee forward progress, this hazard must always be resolved in favor of the instruction that accesses data. What is the total execution time of this instruction sequence in the 5-stage pipeline that only has one memory? We have seen that data hazards can be eliminated by adding nops to the code. Can you do the same with this structural hazard? Why?

4.10.2 [20] <§4.5> For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we change load/store instructions to use a register (without an offset) as the address, these instructions no longer need to use the ALU. As a result, MEM and EX stages can be overlapped and the pipeline has only 4 stages. Change this code to accommodate this changed ISA. Assuming this change does not affect clock cycle time, what speedup is achieved in this instruction sequence?

4.10.3 [10] <§4.5> Assuming stall-on-branch and no delay slots, what speedup is achieved on this code if branch outcomes are determined in the ID stage, relative to the execution where branch outcomes are determined in the EX stage?

4.10.4 [10] <§4.5> Given these pipeline stage latencies, repeat the speedup calculation from 4.10.2, but take into account the (possible) change in clock cycle time. When EX and MEM are done in a single stage, most of their work can be done in parallel. As a result, the resulting EX/MEM stage has a latency that is the larger of the original two, plus 20 ps needed for the work that could not be done in parallel.

4.10.5 [10] <§4.5> Given these pipeline stage latencies, repeat the speedup calculation from 4.10.3, taking into account the (possible) change in clock cycle time. Assume that the latency ID stage increases by 50% and the latency of the EX stage decreases by 10ps when branch outcome resolution is moved from EX to ID.

Pipeline MIPS 2-issue, resolução dos branches no EX, previsão perfeita de saltos e sem delay slot
uma instrução do issue packet pode ser atrasada em relação à outra, mas entram sempre duas instruções de cada vez no pipeline
única restrição no issue packet: não podem ser duas instruções de acesso à memória

1. executar uma iteração do ciclo no código (ie, evolução do pipeline e indicação dos forwardings):

```
        add $2, $8, $0
        sll $3, $8, 2
        add $3, $9, $3
ciclo:  beq $2, $0, fim
        addiu $3, $3, -4
        lw $4, 4($3)
        sw $4, 0($3)
        addi $2, $2, -1
        beq $0, $0, ciclo
fim:
```

corresponde a:

```
        for (i = j; i != 0; i--)
          a[i - 1] = a[i];
```

2. alterar o código para não serem nessários atrasos e repetir

3. calcular o IPC nos dois casos

Última alteração: Segunda, 26 Outubro 2015, 19:45